Documentation for Setting Up Monitoring Stack in EKS for monitoring Worker Nodes & Application

Step 1: Install AWS CLI

AWS CLI allows you to interact with AWS services using the command line.

1. Download AWS CLI v2:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

2. Install Unzip:

```
sudo apt install unzip
```

3. Unzip the AWS CLI archive:

```
unzip awscliv2.zip
```

4. Install AWS CLI:

```
sudo ./aws/install
```

5. Configure AWS CLI:

```
aws configure
```

Provide the following details:

- AWS Access Key ID
- AWS Secret Access Key
- Default region name (e.g., ap-south-1)
- Default output format (click entr)

Step 2: Install Terraform and Clone EKS Terraform Code Repository

Terraform is an open-source tool that allows you to define infrastructure as code.

1. Install Terraform:

```
sudo snap install terraform --classic
```

2. Clone the Terraform EKS repository:

```
git clone https://github.com/jaiswaladi246/EKS-Terraform.git
```

3. Initialize Terraform:
4. cd EKS-Terraform

```
terraform init
```

5. Plan the infrastructure changes:

```
terraform plan
```

6. Apply the infrastructure changes:

```
terraform apply --auto-approve
```

Step 3: Configure Access to EKS Cluster

Once your EKS cluster is created, you need to update your Kubernetes configuration to access it.

1. Update kubeconfig:

```
aws eks --region ap-south-1 update-kubeconfig --name devopsshack-cluster
```

2. Install kubectl for managing Kubernetes clusters:

```
sudo snap install kubectl --classic
```

3. Verify EKS nodes:

```
kubectl get nodes
```

Step 4: Install Helm

Helm is a package manager for Kubernetes. It allows you to deploy, configure, and manage Kubernetes applications easily.

1. Install Helm:

```
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
```

2. Verify Helm installation:

```
helm version
```

Step 5: Configure Prometheus for Monitoring

1. Create values.yml for Prometheus configuration:

```
alertmanager:
  enabled: true
  replicaCount: 1
  statefulSet:
    enabled: true
  persistentVolume:
    enabled: true
    existingClaim: alertmanager-pvc
    accessModes:
      - ReadWriteOnce
    size: 5Gi
    storageClass: manual
server:
  replicaCount: 1
  statefulSet:
    enabled: false
  persistentVolume:
    enabled: false
```

2. Add Prometheus Helm repository:

   ```
   helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
   ```

```
helm repo update
```

3. Install Prometheus using Helm:

   ```
   helm install prometheus prometheus-community/prometheus \
   ```

```
  --namespace monitoring --create-namespace -f values.yml
```

Step 6: Configure Persistent Volume for Prometheus

1. Create a Persistent Volume (PV) configuration pv.yml:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: prometheus-alertmanager-pv
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: ""
  hostPath:
    path: "/mnt/data/prometheus-alertmanager"
```

2. Apply the Persistent Volume configuration:

```
kubectl apply -f pv.yml -n monitoring
```

Step 7: Set Up Load Balancer for Prometheus

1. Edit the Prometheus service to change the service type to LoadBalancer:

```
kubectl edit svc <prometheus-server-svc-name> -n monitoring
```

Step 8: Install Grafana for Visualization

1. Add the Grafana Helm repository:

```
helm repo add grafana https://grafana.github.io/helm-charts
```

```
helm repo update
```

2. Install Grafana:

```
helm install grafana grafana/grafana --namespace monitoring --create-namespace --set adminPassword=admin123
```

3. Edit the Grafana service to change the service type to LoadBalancer:

```
kubectl edit svc grafana -n monitoring
```

Step 9: Configure Blackbox Exporter for Prometheus

1. Install the Blackbox Exporter:

```
helm install blackbox-exporter prometheus-community/prometheus-blackbox-exporter --namespace monitoring --create-namespace
```

2. Edit the Blackbox Exporter service to change the service type to LoadBalancer:

```
kubectl edit svc <black-box-exporter> -n monitoring
```

3. Update the Prometheus configuration to monitor targets via Blackbox Exporter:
    - Download the Prometheus configuration:

```
kubectl get configmap prometheus-server -n monitoring -o yaml > prometheus-configmap.yaml
```

- Add the following job to the Prometheus configuration:

```
- job_name: 'blackbox'
  metrics_path: /probe
  params:
    module: [http_2xx]
  static_configs:
    - targets:
      - https://prometheus.io
      - http://a7594a4e2b8164fc68b353972e345a16-131917163.ap-south-1.elb.amazonaws.com
  relabel_configs:
    - source_labels: [__address__]
      target_label: __param_target
    - source_labels: [__param_target]
      target_label: instance
    - target_label: __address__
      replacement: ab64e10aa10034938bec3b014a3105a9-1105735394.ap-south-
1.elb.amazonaws.com:9115
```

Apply the new configmap file:

```
kubectl apply -f prometheus-configmap.yaml
```

4. Restart Prometheus server pod:

```
kubectl delete pod <prometheus-server-xxxx> -n monitoring
```

Step 10: Set Up Grafana Dashboards

1. Access the Grafana dashboard via the Load Balancer URL.
2. Add Prometheus as a data source:
    - Go to **Connections** > **Data Sources**.
    - Select Prometheus and add the Prometheus Load Balancer URL.
3. Import Grafana dashboards:
    - Click on **Dashboard** > **+** > **Import Dashboard**.
    - Import Node Exporter Dashboard using code from:

https://grafana.com/grafana/dashboards/1860-node-exporter-full/

- Import Prometheus Blackbox Exporter Dashboard using code from:

https://grafana.com/grafana/dashboards/7587-prometheus-blackbox-exporter/

Step 11: Access Prometheus and Grafana

- Access Prometheus using its Load Balancer URL.
- Access Grafana using its Load Balancer URL and login with the admin password you set earlier.